Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Obrstar

Razvojni vgradni sistem BeagleBone Black

diplomsko delo

na univerzitetnem študiju računalništva in informatike

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite strojno zgradbo vgradnega razvojnega sistema BeagleBone Black z mikrokrmilnikom Sitara ARM Cortex-A8 in z operacijskim sistemom Linux Angstrom. Preučite drevo naprav (angl. device tree) v jedru operacijskega sistema. Na razvojni sistem priklopite modul za brezžično komunikacijo Bluetooth. Razvijte vgradno programsko opremo, s katero je možno preko vmesnika Bluetooth krmiliti priključke GPIO na razvojnem sistemu.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Obrstar, z vpisno številko **63040116**, sem avtor diplomskega dela z naslovom:

Razvojni vgradni sistem BeagleBone Black

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 27. septembra 2014 Podpis avtorja:

Zahvaljujem se mentorju izr. prof. dr. Patriciu Buliću za strokovno pomoč in vodenje pri izdelavi diplomske naloge.

Največja zahvala gre očetu Matjažu in mami Andreji za vso podporo in finančno pomoč v času študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
BIOS	Basic input/output system	Osnovni vhodno/izhodni sistem
GPIO	General-purpose input/output	Splošno namenski vhodno izhodni vmesnik
HDMI	High-Definition Multimedia Interface	vmesnik za prenos avdiovizualnih nestisnjenih digitalnih podatkov
I²C	Inter-integrated circuit	Medintegrirano vezje
MUX	Multiplexer	Naprava, ki sestavlja več signalov za prenos po enem vodu
OMAP	Open Multimedia Applications Platform	Odprta večpredstavna aplikacijska platforma
SPI	Serial peripheral interface	Serijski periferni vmesnik
SRM	System Reference Manual	Sistemski referenčni priročnik
SSH	Secure shell	Varna lupina
TRM	Technical Reference manual	Tehnični referenčni priročnik
UART	Universal asynchronous receiver/transmitter	Univerzalni asinhroni sprejemnik/oddajnik
USB	Universal serial bus	Univerzalno serijsko vodilo

Povzetek

Cilj diplomske naloge je omogočiti krmiljenje releja, priključenega na vmesnika GPIO preko brezžičnega vmesnika Bluetooth. V ta namen je potrebno preučiti drevo naprav, z uporabo katerega se nastavi vse potrebno za uporabo vmesnikov GPIO ter UART. Uporaba vmesnika UART je potrebna za priključitev modula Bluetooth. Pred samim programiranjem je bilo posamezne komponente potrebno med seboj povezati in zagotoviti njihovo pravilno delovanje. Izdelan je bil strežniški program, ki sprejema ukaze od odjemalca in po uspešno izvedenih operacijah vrača trenutno stanje releja preko vmesnika Bluetooth. Kot odjemalec je bila uporabljena prenosna naprava, ki jo poganja mobilni operacijski sistem Android, na katerem teče že obstoječa aplikacija za serijsko komunikacijo.

Ključne besede: BeagleBone, razvojni vgradni sistem, ARM, bluetooth, vmesnik GPIO, vmesnik UART.

Abstract

The aim of this thesis is to control a relay connected to the GPIO port over the wireless Bluetooth interface. To achieve this, it is necessary to examine the Device Tree, which is used to enable the use of the GPIO and UART interface. The UART interface is used to connect the Bluetooth module. Before the actual programming the individual components had to be connected and set up properly. A server program was developed, that accepts instructions from the client and returns the current state of the relay after a successful operation via the Bluetooth interface. A portable device, powered by the mobile operating system Android, running an already developed application was used as a client.

Keywords: BeagleBone, embeded development board, ARM, Bluetooth, GPIO, UART.

Poglavje 1

Uvod

Vgrajeni sistemi, zasnovani na družini procesorjev ARM, so v zadnjih letih postali zelo razširjeni. K temu so pripomogli predvsem majhnost naprav, nizka cena in odprtost projektov, katere je z malo truda vsak zanesenjak prilagodil lastnim potrebam. Za velik del priljubljenosti naprav se lahko zahvalimo neprofitni organizaciji Raspberry Pi Foundation. S svojim izdelkom velikosti kreditne kartice Raspberry Pi in viralnim marketingom, so na trgu dosegli velik uspeh. Posledično lahko na svetovnem spletu najdemo ogromno projektov, ki jih poganja računalnik Raspberry Pi. Od enostavnih malih osebnih računalnikov, ki se jih da spraviti v žep, do bolj specifično orientiranih sistemov, kot so termostati, sistemi za zalivanje rož in predvajalniki zabavnih vsebin.

Ena večjih hib sistema Raspberry Pi je zmogljivost, saj mu pri bolj zahtevnih opravilih hitro zmanjka sredstev za delo. Za bolj napredne projekte ali zgolj v želji po večji moči in boljši odzivnosti, lahko tako sežemo po razvojni ploščici BeagleBone Black. Sistem BeagleBone Black je manj razširjen in tudi projektov zanj je zaenkrat še precej malo. Razvoj na sistemu je dosti manj prijazen do uporabnika in je, razen za komericalne razširitve strojne opreme, za uporabo treba poseči globoko v dokumentacijo.

V sklopu diplomske naloge je obravnavan razvojni sistem BeagleBone Black ter opisani osnovni koraki za priklop zunanjih naprav z uporabo dre-

vesa naprav (angl. Device Tree), ki je del Linux jedra (angl. Linux kernel). Praktičen del obsega nastavitve drevesa za delo z vmesnikoma GPIO (splošno namenski vhodno izhodni vmesnik, angl. general-purpose input/output) iter UART (univerzalni asinhroni sprejemnik/oddajnik, angl. universal asynchronous receiver/transmitter) in usposodobitev modula za brezžično komunikacijo Bluetooth.

Poglavje 2

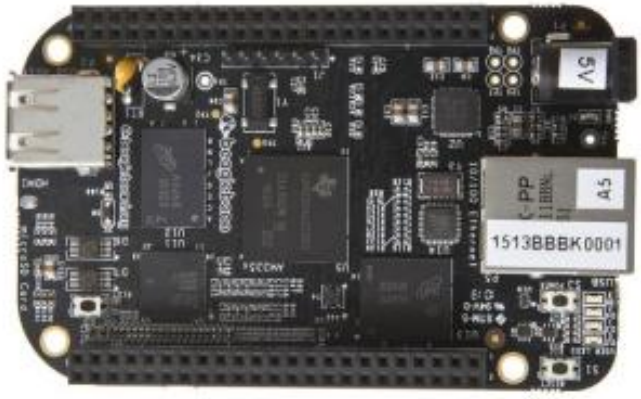
Opis strojne opreme

2.1 Razvojni sistem BeagleBone Black

Bistveni sestavni del diplomskega dela je razvojna ploščica BeagleBone Black, prikazana na sliki 2.1. Opis slednje je povzet po spletni strani razvijalca [8] in uradni spletni aplikaciji wiki [10].

Razvojna ploščica BeagleBone Black je peta izvedenka iz družine razvojnega okolja BeagleBoard podjetja Texas Instruments. Glavne značilnosti ploščice so nizka cena, osredotočenost na visoko razširljivost preko digitalnih nožic (angl. pin) in nizka poraba energije.

Razvojna ploščica, ki jo poganja mikroprocesor Sitara ARM Cortex-A8, ima dva priključka: P8 in P9. Vsak od priključkov ima 46 digitalnih nožic (skupno 92) s toleranco 3,3V. Skozi nožice lahko teče le šibek elektični tok, od 4mA do 6mA za izhod in do 8mA za vhod. Uporabniku je za uporabo privzeto na voljo 32 digitalnih nožic, kar lahko povečamo na 69 z izklopom drugih funkcionalnosti. V ta namen lahko izključimo izhod HDMI (vmesnik za prenos avdiovizualnih nestisnjenih digitalnih podatkov, angl. High-Definition Multimedia Interface), bralec spominskih kartic, ... Razlog za tem je multipleksiranje digitalnih nožic, ki se jih da vypostaviti v enega izmed sedmih različnih načinov delovanja. Za priklop zunanjih naprav so predvsem zanimivi vmesniki GPIO, UART, I^2C (medintegrirano vezje, angl.



Slika 2.1: Razvojni sistem BeagleBone Black [7].

inter-integrated circuit) in SPI (serijski periferni vmesnik, angl. serial peripheral interface). Razvojna ploščica omogoča napajanje zunanjih naprav z napetostjo 3,3V ali 5V.

Ploščica se napaja preko 5V prilagojevalca ali priključka USB (univerzalno serijsko vodilo, angl. universal serial bus). Na razvojno ploščico se povežemo preko ožičenega omrežja (priključek RJ45) ali priključka USB, do katere dostopamo preko protokola SSH (varna lupina, angl. secure shell). Razvojno ploščico lahko preko izhoda HDMI povežemo z zunanjim zaslonom in nanj preko priključka USB priklopimo tipkovnico in miš.

Privzeto ploščico poganja operacijski sistem GNU/Linux, natančneje izvedba Armstrong z jedrom 3.8, ki je že nameščen in vsebuje vsa potrebna orodja za razvoj na ploščici. Z nekaj truda lahko namestimo tudi katero izmed preostalih izvedb, npr.: Gentoo, ArchLinux, Beaglenmt ... ali celo mobilni operacijski sistem Android.

2.1.1 Povezovanje z razvojnim okoljem

Če ploščico povežemo z zunanjim zaslonom, tipkovnico in miško, nas po zagonu pričaka namizno okolje GNOME, kjer lahko takoj pričnemo z delom.

Alternativno nam ostaneta še povezava preko ožičenega omrežja ali priključka USB. V obeh primerih za dostop do naprave uporabimo enega izmed odjemalcev SSH (openssh, Putty, ...). Prijavimo se kot korenski uporabnik (angl. root) z ukazom 2.1, geslo pa pustimo prazno.

```
# ssh -l root 192.168.7.2
```

Ukaz 2.1: za vzpostavitev povezave SSH.

Operacijskem sistemu GNU/Linux prepozna razvojno ploščico, povezano preko priključka USB, če jedro vključuje nastavitve 2.2.

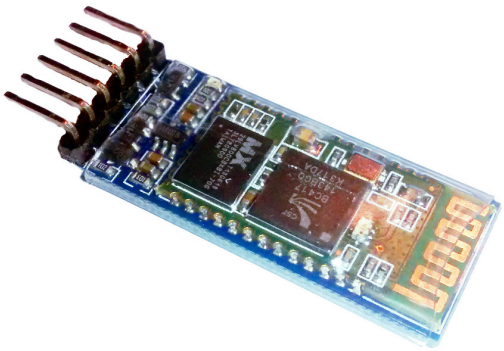
```
Device Drivers  --->
[*] Network device support  --->
    USB Network Adapters  --->
    <*> Multi-purpose USB Networking Framework
    <*>   CDC Ethernet support
          (smart devices such as cable modems)
    <*>   Host for RNDIS and ActiveSync devices
```

Nastavitve 2.2: Linux jedra, ki omogočajo prepoznavo razvojne ploščice BeagleBone Black.

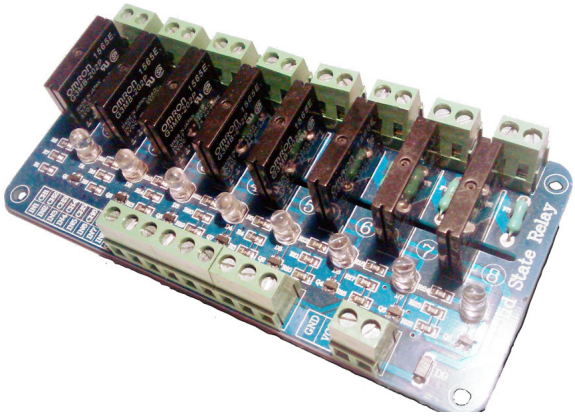
2.2 Modul Bluetooth in rele

Dve izmed razpoložljivih digitalnih nožic smo porabili za priklop modula Bluetooth, prikazanega na sliki 2.2. Ta s ploščico komunicira preko vmesnika UART in se napaja preko 3,3V izhoda na razvojni ploščici. Modul je vrste suženj, kar pomeni, da sprejema povezave od zunanjih naprav, sam pa jih ne more vzpostavljati.

Na sliki 2.3 je prikazan polprevodni 8-kanalni rele, ki smo ga priklopili na osem digitalnih nožic. Napaja se preko 5V izhoda na ploščici. Kanal na releju je izklopljen oz. v nizkem stanju, kadar ima vhodno napetost med 0V in 0,5V, ter vklopljen oz. v visokem stanju, kadar je vhodna napetost



Slika 2.2: Brezžični modul Bluetooth.



Slika 2.3: 8-kanalni 5V polprevodni rele.

med 2,5V in 20V. Napetosti so primerne za ploščico, ki ima na nožicah nizko stanje 0V in visoko stanje 3,3V.

Poglavje 3

Drevo naprav

Procesorji ARM v nasprotju z osebnimi računalniki in strežniki nimajo nabora osnovnih podprogramov BIOS (osnovni vhodno/izhodni sistem, angl. basic input/output system), ki bi skrbel za poenoten način predstavitve strojne opreme operacijskemu sistemu. Zato so morali proizvajalci teh naprav jedru dodati lastno kodo za vsako napravo. S priljubljenostjo in razširitvijo procesorjev ARM je to postalo neprakitčno, saj je velikost jedra strmno naraščala. Da bi se temu ognili, je bilo z jedrom različice 3.7 uvedeno drevo naprav (angl. Device Tree). Pred tem se je uporabljala aplikacijska platforma OMAP (odprta večpredstavna aplikacijska platforma, angl. Open Multimedia Applications Platform) MUX (multiplekser, angl. multiplexer).

Drevo naprav [4] je podatkovna stuktura, ki opisuje strojno opremo. Namesto da vse podrobnosti o napravi vprogramiramo neposredno v operacijski sistem, lahko večino opišemo v podatkovni strukturi, ki jo nato posredujemo operacijskemu sistemu med zagonom.

3.1 Opis drevesa naprav

Datoteke z izvorno kodo, ki opisujejo drevo naprav, imajo končnico `.dts`. Prevedemo jo s prevajalnikom dtc (prevajalnik drevesa naprav, angl. Open Firmware device-trees compiler). Izvorna koda drevesa se prevede v binarno

datoteko s končnico `.dtbo` (prekrivalna datoteka drevesa naprav, angl. device tree blob overlay), ki jo lahko naložimo, odstranimo ali zamenjamo kar med samim delovanjem operacijskega sistema. To je med razvojem zelo priročno, saj nam ob vsaki spremembi ni treba ponovno zagnati operacijskiega sistema. Možno je nastaviti samodejno nalaganje prekrivalnih datotek, da nam jih ni treba ročno naložiti po vsakem novem zagonu.

3.2 Razčlemba izvorne kode `.dts`

Drevo naprav je preprosta struktura vozlišč (angl. nodes) in lastnosti naprave. Lastnosti sta dvojčka: ključ in vrednost. Vozlišče poleg teh lastnosti lahko vsebuje tudi druga vozlišča [9].

V nadaljevanju so izseki izvorne kode `.dts` [6], ki prikazuje vklop vmesnika UART1 na razvojni ploščici BeagleBone (Black). Z drugimi besedami - program nastavi nožici 24 in 26 na priključku P9 za delovanje v načinu UART.

Odrezek kode 3.1 nam pove različico datoteke `.dts` ter da gre za vtičnik, delček 3.2 pa označuje začetek korenskega vozlišča.

```
/dts-v1/;
/plugin/;
```

Koda 3.1: Prvi odrezek kode za vklop vmesnika UART1.

```
/ {
```

Koda 3.2: Začetek korenskega vozlišča.

Del kode 3.3 pove za katere naprave je drevo namenjeno. V tem primeru sta to razvojni ploščici BeagleBone in BeagleBone Black podjetja Texas Instruments.

```
compatible = "ti,beaglebone", "ti,beaglebone-black";
```

Koda 3.3: Del kode, ki našteje združljive naprave.

Sklopa "part-number" in "version" v odrezku kode 3.4 zagotavljata, da se naloži pravilno drevo. Poleg tega se oba podatka uporabljata tudi kot konvencija poimenovanja datoteke `.dtbo`.

```
/* identification */
part-number = "enable-UART1";
version = "00A0";
```

Koda 3.4: Odrezek določi ime izhodne datoteke `enable-UART1-00A0.dtbo`

Koda 3.5 vsebuje drobec drevesa naprav (angl. device tree fragment). Drobec "fragment@0" opisujej tarčo, ki jo želimo prekriti ter vsebuje nastavitve digitalnih nožic.

V našem primeru želimo prekriti tarčo "am33xx_pinmux". Tarča je združljiva z gonilnikom "pinctrl-single" [5], ki nastavi način delovanja podani digitalni nožici.

Vrstici "enable_UART1: pinmux_enable_uart1_2_pins", ki sledi začetku prekrivalnega bloka "__overlay__", lahko dodelimo poljubno ime. V kodnem bloku "pinctrl-single", nastavimo način delovanja želenim digitalnim nožicam. Sledita dvojčka: odmik za naslavljanje želenih digitalnih nožic 0x180 (digitalna nožica P9_26) in 0x184 (P9_24) ter vrednost želenega načina delovanja za obe nožici 0x20 in 0x00 (za bolj podrobno razlago glej poglavje 4.1).

```
fragment@0 {
  target = <&am33xx_pinmux>;
  __overlay__ {
    enable_UART1: pinmux_enable_uart1_2_pins {
      pinctrl-single,pins = <
        /*P9_26 uart1_txd.uart1_txd MODE0 INPUT (RX)*/
        0x180 0x20
        /*P9_24 uart1_rxd.uart1_rxd MODE0 OUTPUT (TX)*/
        0x184 0x00
      >;
    };
```

```
    };
};
```

Koda 3.5: Drobec fragment@0 nastavi digitalni nožici

Podobno kot prej, imamo tudi v izseku 3.6 drobec "fragment@1", vendar tarčo "am33xx_pinmux" nadomesti tarča "uart2", ker želimo vklopiti vmesnik UART in ne nastaviti digitalne nožice. Spremenljivka "pinctrl-0" se sklicuje na lasnosti "enable_UART1" iz drobca "fragment@0". S prvim drobecem "fragment@0" nastavimo način delovanja digitalnih nožic, s spremenljivko "pinctrl-0" v drobcu "fragment@1" pa nožice dodelimo napravi UART.

```
fragment@1 {
    target = <&uart2>;      /* really uart1 */
    __overlay__ {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&enable_UART1>;
    };
};
```

Koda 3.6: Drobec fragment@1 dodeli nožice napravi UART

3.3 Prekrivanje

Kot smo omenili, je med razvojem najbolj elegantno uporabljati prekrivanje. Datoteko `en-GPIO.dts` z izvorno kodo prevedemo z ukazom 3.7.

```
# dtc -O dtb -o en-GPIO-00A0.dtbo -b 0 -@ en-GPIO.dts
```

Ukaz 3.7: Prevede izvorno kodo drevesa naprav.

Dobljeno binarno datoteko posnamemo v mapo `/lib/firmware` in jo naložimo z ukazom 3.8.

```
# echo en-GPIO > /sys/devices/bone_capemgr.8/slots
```

Ukaz 3.8: Naloži prekrivanje.

Katera prekrivanja so trenutno v uporabi lahko ugotovimo z branjem datoteke `slots`.

```
# cat /sys/devices/bone_capemgr.8/slots
```

Ukaz 3.9: Prebere datoteko `slots`.

Ukaz 3.9 nam vrne nekaj podobnega, kot prikazuje izpis 3.10.

```
0: 54:PF---
1: 55:PF---
2: 56:PF---
3: 57:PF---
4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,
   BB-BONE-EMMC-2G
5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,
   BB-BONELT-HDMI
6: ff:P-O-L Override Board Name,00A0,Override Manuf,
   en-GPIO
7: ff:P-O-L Override Board Name,00A0,Override Manuf,
   en-UART
```

Izpis 3.10: Vsebina datoteke `slots`.

Če se zmotimo pri pisanju naše kode ali imamo za to kak drug razlog, lahko določeno prekrivanje, s poznanjem njegovega zaporednega števila, tudi odstranimo. Prekrivanje "en-GPIO" odstanimo z ukazom 3.11:

```
# echo -6 > /sys/devices/bone_capemgr.8/slots
```

Ukaz 3.11: Zapiše število "-6" v datoteko `slots`.

Ta način trenutno ni najbolj zanesljiv in lahko privede do nepričakovanjega obnašanja ali celo do nestabilnega sistema. Namesto odstranjanja prekrivanja je priporočeno razvojno ploščico raje ponovno zagnati.

Za samodejno nalaganje prekrivanja v `/media/BEAGLEBONE/uEnv.txt` dodamo "capemgr.enable_partno=" in ime prekrivanja. Če dodamo prekrivanje `en-UART`, je vsebina `/media/BEAGLEBONE/uEnv.txt` enaka izpisu 3.12.

```
optargs=quiet capemgr.enable_partno=en-UART
```

Izpis 3.12: Vsebina datoteko `uEnv.txt`.

Trenutno to deluje samo, če spremenimo nastavitev privzetega jedra in ga na novo prevedemo ali ga nadomestimo z novim, ki že ima vse potrebne nastavitve.

Poglavje 4

Upravljanje vmesnika GPIO

Za tem, ko nastavimo vse potrebno za delovanje digitalnih nožic v načinu GPIO, jih lahko začnemo uporabljati. Na začetku deluje kot izhod in se nahajajo v nizkem izhodnem stanju. Preden lahko začnemo delati z željeno digitalno nožico GPIO jo moramo omogočiti. To storimo tako, da število željene nožice GPIO zapišemo v datoteko `/sys/class/gpio/export`. Števila nožice GPIO najdemo v stolpcu "MODE7" tabel "Expansion Header P8 Pinout" in "Expansion Header P9 Pinout" SRM (sistemski referenčni priročnik, angl. System Reference Manual) razvojne ploščice [2]. Podana so v obliki gpioX[Y], kar pretvorimo v numerično vrednosti po formuli (4.1) (izračun za primer gpio2[3] (4.2)).

$$X * 32 + Y \tag{4.1}$$

$$2 * 32 + 3 = 67 \tag{4.2}$$

Nožico GPIO omogočimo z ukazom 4.1.

```
# echo -n 67 > /sys/class/gpio/export
```

Ukaz 4.1: Zapiše število "67" v datoteko `export`.

Pojavi se nova mapa `/sys/class/gpio/gpioX`, kjer je X zaporedno število omogočene digitalne nožice GPIO. Mapa vsebuje 5 datotek in 2 mapi, od teh sta zanimivi predvsem naslednji datoteki:

- `direction` - vsebina pove ali digitalna nožica GPIO deluje kot vhod (vsebuje vrednost "in") ali kot izhod (vsebuje vrednost "out").

- `value` - vsebina pove ali je digitalna nožica GPIO v nizkem (vsebuje vrednost "0") ali v visokem stanju (vsebuje vrednost "1").

Obe datoteki lahko preberemo, za pridobitev trenutnega stanje ali vanj zapišemo novo željeno stanje. V datoteko `value` lahko uspešno pišemo le, če nožica GPIO deluje kot izhod.

Ko nožice GPIO več ne potrebujemo jo lahko onemogočimo z zapisom njeno zaporedno število v datoteko `/sys/class/gpio/unexport`.

```
# echo -n 67 > /sys/class/gpio/unexport
```

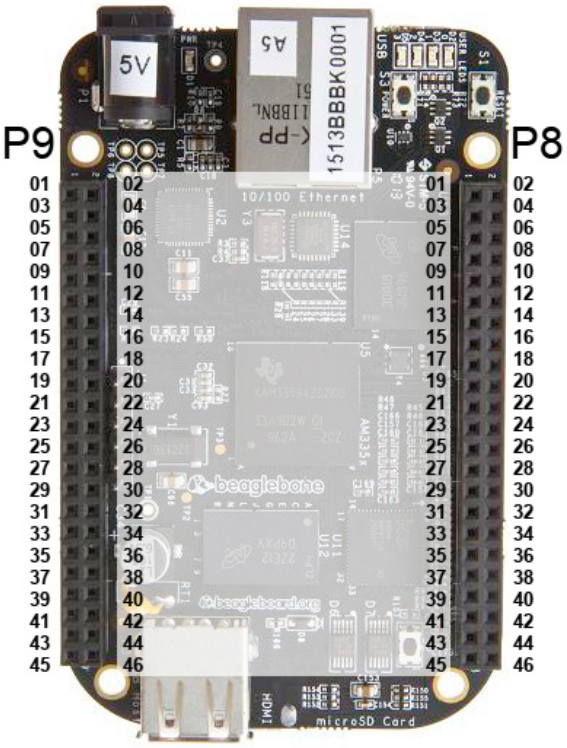
Ukaz 4.2: Zapiše "67" v datoteko `unexport`.

Z ukazom 4.2 izgubimo vse dosedanje nastavitve, digitalna nožica GPIO pa se povrne v privzeto nizko stanje.

4.1 Naslavljanje digitalnih nožic

Kot smo videli v poglavju 3.2, za naslavljanje digitalnih nožic in nastavljanje načina njihovega delovanja uporabljamo dvojčka vrednosti: naslov z odmikom in vrednost, ki določa način delovanja. Naslov želene digitalne nožice najdemo s pomočjo SRM razvojne ploščice BeagleBone Black [2] in TRM (tehnični referenčni priročnik, angl. Technical Reference Manual) mikroprocesorja Sitara ARM Cortex-A8 [1]. Za želeno digitalno nožico moramo vedeti njeno zaporedno število in na katerem priključku se nahaja (P8 ali P9), kar lahko razberemo is slike 4.1.

Izbrano digitalno nožico poiščemo v tabeli "Expansion Header P8 Pinout" ali "Expansion Header P9 Pinout" in si zabeležimo ime v stolpcu "MODE0". V TRM mikroprocesorja se v tabeli "CONTROL_MODULE Registers" nahajajo odmični naslovi registrov od naslova 44e10000 dalje, ki jih potrebujemo za naslavljanje nožic. Ime iz tabele "Expansion Header Pinout" ima svoj



Slika 4.1: Razporeditev digitalnih nožic na razvojni ploščici BeagleBone Black.

par v tabeli "CONTROL_MODULE Registers", s to razliko da ima v slednji predpono "conf_". Dobljenemu odmiku moramo odšteti 800, ker so v TRM odmiki od naslova 44e10000 dalje, jedro pa odmik prišteje naslovu 44e10800. Tabela 4.1 prikazuje vse odmične naslove za priključka P8 in P9.

Primer: digitalna nožica na priključku P8 z zaporednim številom 3 ima v stolpcu "MODE0" vrednost "gpmc_ad6", kar sovpada z vrednostjo v TRM mikroprocesorja "conf_gpmc_ad6". Vidimo, da je odmični naslov 818, čemur odštejemo 800. Dobimo odmik 0x018 za digitalno nožico P8_3.

digitalna nožica	**odmik za P8**	**odmik za P9**	**GPIO za P8**	**GPIO za P9**
1	/	/	/	/
2	/	/	/	/
3	0x018	/	gpio38	/
4	0x01C	/	gpio39	/
5	0x008	/	gpio34	/
6	0x00C	/	gpio35	/
7	0x090	/	gpio66	/
8	0x094	/	gpio67	/
9	0x09C	/	gpio69	/
10	0x098	/	gpio68	/
11	0x034	0x070	gpio45	gpio30
12	0x030	0x078	gpio44	gpio60
13	0x024	0x074	gpio23	gpio31
14	0x028	0x048	gpio26	gpio50
15	0x03C	0x040	gpio47	gpio48
16	0x038	0x04C	gpio46	gpio51
17	0x02C	0x15C	gpio27	gpio5
18	0x08C	0x158	gpio65	gpio4
19	0x020	0x17C	gpio22	gpio13
20	0x084	0x178	gpio63	gpio12
21	0x080	0x154	gpio62	gpio3
22	0x014	0x150	gpio37	gpio2
23	0x010	0x044	gpio36	gpio49
24	0x004	0x184	gpio33	gpio15
25	0x000	0x1AC	gpio32	gpio117
26	0x07C	0x180	gpio61	gpio14
27	0x0E0	0x1A4	gpio86	gpio115
28	0x0E8	0x19C	gpio88	gpio113
29	0x0E4	0x194	gpio87	gpio111

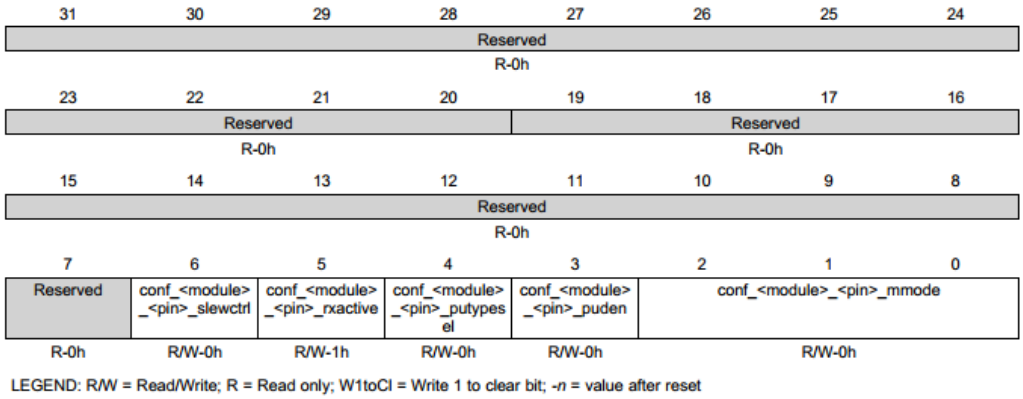
30	0x0EC	0x198	gpio89	gpio112
31	0x0D8	0x190	gpio10	gpio110
32	0x0DC	/	gpio11	/
33	0x0D4	/	gpio9	/
34	0x0CC	/	gpio81	/
35	0x0D0	/	gpio8	/
36	0x0C8	/	gpio80	/
37	0x0C0	/	gpio78	/
38	0x0C4	/	gpio79	/
39	0x0B8	/	gpio76	/
40	0x0BC	/	gpio77	/
41	0x0B0	0x1B4	gpio74	gpio20
41	/	0x1A8	/	gpio116
42	0x0B4	0x164	gpio75	gpio7
42	/	0x1A0	/	gpio114
43	0x0A8	/	gpio72	/
44	0x0AC	/	gpio73	/
45	0x0A0	/	gpio70	/
46	0x0A4	/	gpio71	/

Tabela 4.1: Odmični naslovi za digitalne nožice.

4.2 Načini delovanja digitalnih nožic

Digitalnim nožicam nastavimo želen način delovenja s pisanjem v register conf_<module>_<pin> (Slika 4.2). Za nas je zanimivih spodnjih 7 bitov, saj so ostali rezervirani:

- bit 6 - conf_<module>_<pin>_slewctrl - določi hitro (vrednost $0_{(2)}$) ali počasno (vrednost $1_{(2)}$) strmino signala (angl. slew rate).
- bit 5 - conf_<module>_<pin>_rxactive - določi ali digitalna nožica de-



Slika 4.2: Register conf_<module>_<pin> [1].

luje le kot izhod (vrednost $0_{(2)}$) ali tudi kot vhod (vrednost $1_{(2)}$).

- bit 4 - conf_<module>_<pin>_putypesel - določi ali je nizka vhodna napetost nizko (angl. pullup) (vrednost $1_{(2)}$) ali visoko vhodno stanje (angl. pulldown) (vrednost $0_{(2)}$).

- bit 3 - conf_<module>_<pin>_puden - določi ali je notranji pullup/pulldown upor vklopljen (vrednost $0_{(2)}$) ali izklopljen (vrednost $1_{(2)}$).

- bit 2-0 - conf_<module>_<pin>_mode - določi načina delovanja:

  - MODE0 - vrednost $000_{(2)}$
  - MODE1 - vrednost $001_{(2)}$
  - MODE2 - vrednost $010_{(2)}$
  - MODE3 - vrednost $011_{(2)}$
  - MODE4 - vrednost $100_{(2)}$
  - MODE5 - vrednost $101_{(2)}$
  - MODE6 - vrednost $110_{(2)}$
  - MODE7 - vrednost $111_{(2)}$

Različni načini delovanja (od MODE0 do MODE7) so našteti v SRM razvojne ploščice v tabeli "Expansion Header P8 Pinout" ali "Expansion Header P9 Pinout" [2]. Končna vrednost so vse želene vrednosti v šestnajtiški obliki.

Primer: Želimo, da digitalna nožica sprejema ($1_{(2)}$) signal v pullup ($1_{(2)}$) MODE7 ($111_{(2)}$) načinu s hitro strmino signala ($0_{(2)}$). Ker želimo, da deluje v pullup načinu, moramo nastaviti uporabo notranjega upora ($0_{(2)}$). Končna vrednost je $001101111_{(2)} = 0x37_{(16)}$.

Vsebina datoteke `/sys/kernel/debug/pinctrl/44e10800.pinmux/pins` vsebuje nastavitve načina delovanja digitalnih nožic, kot prikazuje izpis 4.3. Zanimata nas drugi in tretji stolpec, iz katerih razberemo naslov digitalne nožice in način delovanja (vrednost spodnjih 7 bitov, ki jih zapišemo v register conf_<module>_<pin>).

```
registered pins: 142
pin 0 (44e10800) 00000031 pinctrl-single
pin 1 (44e10804) 00000031 pinctrl-single
...
pin 35 (44e1088c) 00000027 pinctrl-single
pin 36 (44e10890) 00000007 pinctrl-single
pin 37 (44e10894) 00000007 pinctrl-single
pin 38 (44e10898) 00000007 pinctrl-single
```

Izpis 4.3: Vsebina datoteke `pins`.

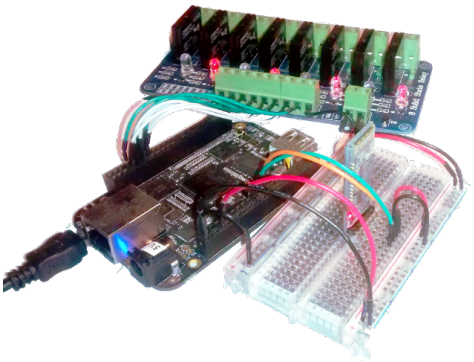
Poglavje 5

Opis celotnega sistema in aplikacije

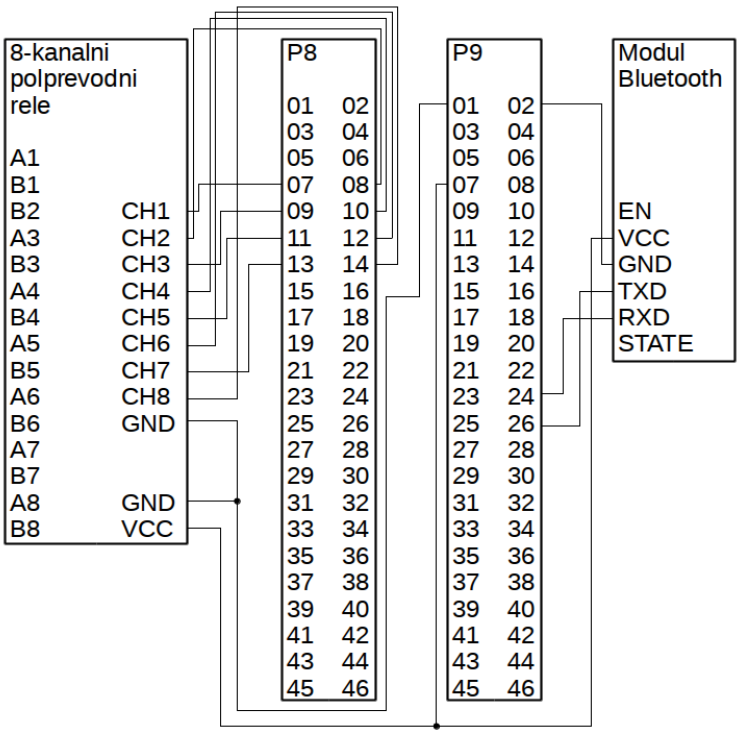
Celotni sistem (slika 5.1) sestavljajo tri komponente, opisane v poglavju 2:

- razvojna ploščica BeagleBone Black
- brezžični modul Bluetooth
- 8-kanalni 5V polprevodni rele

Na sliki 5.2 si lahko ogledamo shematiko vseh komponent povezanih v celoto. Tako modul Bluetooth kot tudi rele sta povezana na napajanje in digitalne



Slika 5.1: Celoteni sistem.



Slika 5.2: Shematika celotnega sistema.

nožice razvojne ploščice. Tabela 5.1 prikazuje medsebojno povezanost komponent in v katerem načinu delujejo uporabljene digitalne nožice na razvojni ploščici.

Digitalna nožica na razvojni ploščicik	**Način delovanja**	**Priključek na modulu Bluetooth**	**Priključek na releju**
P8_7 (gpio66)	0x07	/	CH1
P8_8 (gpio67)	0x07	/	CH2
P8_9 (gpio69)	0x07	/	CH3
P8_10 (gpio68)	0x07	/	CH4
P8_11 (gpio45)	0x07	/	CH5
P8_12 (gpio44)	0x07	/	CH6
P8_13 (gpio23)	0x07	/	CH7
P8_14 (gpio26)	0x07	/	CH8

P9_24 (UART1_TXD)	0x00	RXD	/
P9_26 (UART1_RXD)	0x20	TXD	/
P9_7 (VDD_5V)	/	VCC	VCC
P9_1/2 (GND)	/	GND	GND

Tabela 5.1: Povezave med komponentami.

5.1 Osnovne nastavitve in prekrivanje

Nastavitve iz tabele 5.1 so potrebne za pravilno delovanje programa. Vseh osem digitalnih nožic na priključku P8 (od 7 do 14) deluje v načinu MODE7 (vmesnik GPIO) in delujejo izključno kot izhod. Uporabljajo se za upravljanje kanalov na polprevodnem releju.

Digitalni nožici na priključku P9 obe delujeta v načinu MODE0 (vmesnik UART). Digitalna nožica 24 deluje kot izhod, saj preko nje pošiljamo podatke brezžičnem modulu Bluetooth, digitalna nožica 26 pa kot vhod, ki podatke od modula Bluetooth sprejema.

Načine delovanja nastavimo z dvema prekrivanjema drevesa naprav:

- `enable-GPIO.dtbo` - nastavi vse potrebno za pravilno delovanja digitalnih nožic na priključku P8 in omogoči njuno uporabo, kot je prikazano v poglavju 4
- `enable-UART.dtbo` - nastavi vse potrebno za komunikacijo med razvojno ploščico in modulom Bluetooth preko vmesnika UART na priključku P9. Ustvari datoteko naprave `/dev/ttyO1`, ki jo uporabimo za komunikacijo z modulum Bluetooth.

5.1.1 Nastavitve modula Bluetooth

Ploščica z modulom pridobi sposobnost brezžične komunikacije preko protokola Bluetooth. Pred prvo uporabo moramo določiti nekaj osnovnih nastavitev, za kar imamo na voljo naslednje ukaze [3]:

- AT - preizkus povezave. Če je povezave vspostavljena vrne “OK”.

- AT+BAUDx - hitrost prenosa, izražena v baudih. V našem primeru AT+BAUD4 oz. 9600bps. Če je hitrost nastavljena uspešno vrne “OK9600” oz. OK, čemur sledi željena nastavljena hitrost.

- AT+PINxxxx - nastavimo štiri mestno numerično geslo. Ob uspešni nastavitvi gesla vrne “OKsetPIN”.

- AT+NAMExxxx - nastavimo ime pod katerim se modul Bluetooth predstavi drugim napravam. Ob uspešni nastavitvi imena vrne “OKsetname”.

Ukaze modulu posredujemo preko datoteka naprav `/dev/tty01`, za kar lahko uporabimo enega izmed programov za serijsko komunikacijo (minicom, ZOC, ...). V datoteko `/dev/tty01` lahko pišemo tudi v lastnem programu ali kar neposredno iz GNU/Linux lupine z ukazom 5.1 (potrebno je uporabiti stikalo “-n”, da ne pošljemo tudi znaka za novo vrstico, ker ga vmesnik UART obravnava kot vsak drug znak).

```
# echo -n AT > /dev/tty01
```

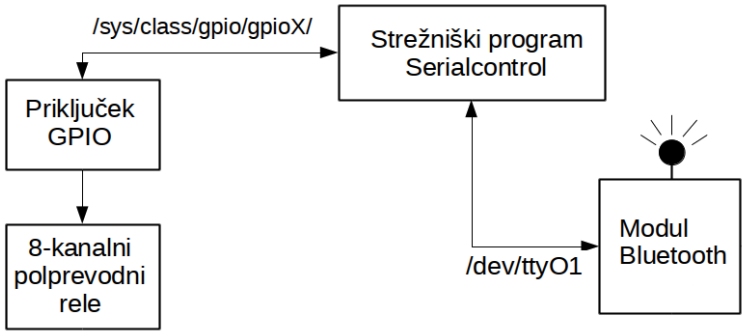
Ukaz 5.1: Zapiše zaporedje znakov “AT” v datoteko `tty01`

Pravilnost delovanja preverimo z branjem `/dev/tty01` (minicom to počne samodejno, lahko pa uporabimo ukaz 5.2):

```
# cat /dev/tty01
```

Ukaz 5.2: Izpiše vsebino datoteko `tty01`

V primeru, da ima modul Bluetooth vzpostavljeno povezavo z zunanjo napravo, se vsi znaki, ki jih pošljemo datoteki naprav `/dev/tty01` zaporedno posredujejo povezani napravi.



Slika 5.3: Diagram komunikacije med programom in napravami.

5.2 Strežnik

Sistem deluje na principu strežnik-odjemalec. Program `Serialcontrol`, ki teče na razvojni ploščici BeagleBone Black, prevzame vlogo strežnika in čaka na zahteve od zunanje naprave, ki se nanj poveže preko modula Bluetooth. Glede na prejete zhatevke program preko nožic GPIO vklaplja in izklaplja kanale na releju. Komunikacijo med posameznimi sestavnimi deli sistema prikazuje slika 5.3.

5.3 Odjemalec

Za serijsko komunikacijo preko protokola Bluetooth smo uporabili aplikacijo "Bluetooth spp tools pro" (dostopno na: `https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO&hl=en`) s prilagojenim uporabniškim vmesnikom (slika 5.4). Ko pritisnemo na gumb v aplikaciji, ta pošlje zaporedje treh znakov modulu Bluetooth, ki jih preko protokola UART nadalnjo posreduje strežnemu programu `Serialcontrol`. `Serialcontrol` nato glede na prejeto zaporedje znakov izvede eno izmed naslednjih operacij iz tabele 5.2.

Po opravljeni operaciji strežnik `Serialcontrol` vrne trenutno stanje re-

leja v obliki zaporedja ničel in enic. Stanje se izpiše na zaslonu odjemalca, kot prikazuje izpis 5.3.

Gumb	**Poslano zaporedje**	**Izvršen ukaz**
toggle	7gl	Spremeni trenutno stanje kanalov na releju
1-8	tgX, $1 \leq X \leq 8$	Vklopi/izklopi kanal X na releju
On	1on	Izklopi vse kanale na releju
Off	2of	Vklopi vse kanale na releju
quit	3gl	Izhod iz programa `Serialcontrol`

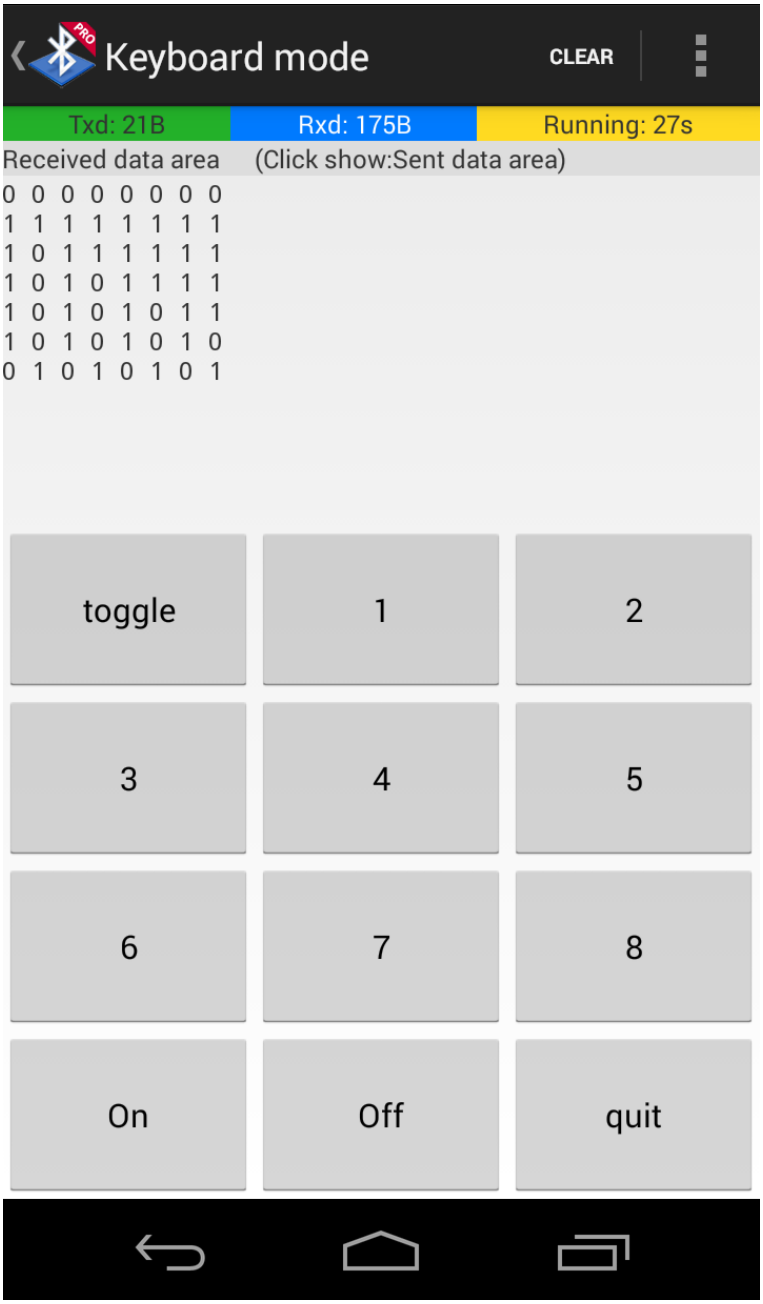
Tabela 5.2: Zaporedje prejetih znakov in ustrezne operacije.

```
/* Vklopljena sta drugi in peti kanal releja */
0 1 0 0 1 0 0 0
```

Izpis 5.3: Primer izpisa v uporabniškem vmesniku odjemalca.

Števila predstavljajo trenutno stanje kanalov releja in si sledijo v naraščajočem vrstnem redu:

- 0 - kanal je izklopljen.
- 1 - kanal je vklopljen.



Slika 5.4: Uporabniški vmesnik aplikacije "Bluetooth spp tools pro" za komunikacijo preko protokola Bluetooth.

Poglavje 6

Sklepne ugotovitve

V diplomski nalogi so bili opisani in preučeni osvnoni sestavni deli sistema za krmiljenje zunanjih naprav priključenih na vmesnik GPIO preko brezžičnega vmesnika Bluetooth. Poleg tega je bilo v obsegu naših potreb preučeno drevo naprav ter njegove osnovne nastavitve. Končen rezultat je delujoč sistem s programsko opremo za krmiljenje releja preko vmesnika Bluetooth s pomočjo mobilnega operacijskega sistema Android, ki ga poganja pametni mobilni telefon ali tablica. Med najbolj časovno zamudne dele spada pridobitev odmičnih naslovov za naslavljanje digitalnih nožic, ki so prikazani v tabeli 4.1.

Z izdelanim sistemom je mogoče brezžično krmiliti do osem porabnikov elektične energije, število katerih je enostavno povečati z nekaj manjšimi spremembami. Med možne razširitve sodijo dodatni načini upravljanje naprav na vmesniku GPIO (recimo preko spleta ali mobilnega omrežja), priklop porabnikov električne energije, ter uporaba vmesnika GPIO kot vhoda za pridobivanje bolj ažurnih podatkov o stanju releja in nanj priklopljenih porabnikov. Vsekakor pa bi pred praktično uporabo sistema bilo potrebno bolje zavarovati brezžično komunikacijo med odjemalcem in samim sistemom.

Nadaljnje delo vključuje dodatno preučevanje drevesa naprav za uporabo ostalih vmesnikov na razvojni ploščici BeagleBone Black, kar je s sedanjim poznavajem odmičnih naslovov digitalnih nožic olajšano. Posledično bi bilo

mogoče na sistem priključiti veliko število drugih naprav, kar dodatno razširi možnosti za nadaljnji razvoj.

Literatura

[1] AM335x Sitara Technical Reference Manual, dostopno na:
`http://www.ti.com/lit/ug/spruh73k/spruh73k.pdf`

[2] BeagleBone Black System Reference Manual, dostopno na:
`https://github.com/CircuitCo/BeagleBone-Black/blob/master/BBB_SRM.pdf?raw=true`

[3] Bluetooth Slave UART Board Wireless Module Transceiver Manual, dostopno na:
`http://www.wvshare.com/downloads/accBoard/Bluetooth-UART-Board.7z`

[4] Definicija drevesa naprav, dostopno na:
`http://devicetree.org/`

[5] Opis pinctrl gonilnika, dostopno na:
`https://www.kernel.org/doc/Documentation/devicetree/bindings/pinctrl/pinctrl-single.txt`

[6] Razčlemba `.dts` datoteke, dostopno na:
`https://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree/device-tree-overlays`

[7] Spletna stran proizvajalca razvojne ploščice BeagleBone Black, dostopno na:
`http://www.ti.com/tool/beaglebk`

[8] Spletna stran razvojne ploščice BeagleBone Black, dostopno na:
`http://beagleboard.org/Products/BeagleBone+Black`

[9] Uporaba drevesa naprav, dostopno na:
`http://devicetree.org/Device_Tree_Usage`

[10] Uradna spletna aplikacija wiki razvojne ploščice BeagleBone Black, dostopno na:
`http://elinux.org/Beagleboard:BeagleBoneBlack`